Refounding Software Engineering: The Semat Initiative

Mira Kajko-Mattsson, Ivar Jacobson, Brian Elvesæter, Michael Goedicke
Agenda

Semat Presentation

- The Semat Initiative
  - Ivar Jacobson
- The Semat Kernel
  - Mira Kajko-Mattsson
- The Language
  - Michael Goedicke
- Evaluation of Semat
  - Brian Elvesæter
- The Value of Semat
  - Ivar Jacobson

Semat Panel

- Participants
  - The Semat members
  - Bertrand Meyer
  - Barry Boehm
- Subject:
  - Do we need Semat?
Agenda

Semat Presentation

- The Semat Initiative
  - Ivar Jacobson
- The Semat Kernel
  - Mira Kajko-Mattsson
- The Language
  - Michael Goedicke
- Evaluation of Semat
  - Brian Elvesæter
- The Value of Semat
  - Ivar Jacobson

Semat Panel

- Participants
  - The Semat members
  - Bertrand Meyer
  - Barry Boehm
- Subject:
  - Do we need Semat?
The Semat initiative

SOFTWARE ENGINEERING METHOD AND THEORY

www.semat.org

Founded by the Troika in Sep 2009:
Ivar Jacobson
Bertrand Meyer
Richard Soley
…we look like a fashion industry

Software Development is driven by fashions and fads

- Twenty years ago it was all about OO
- Twelve years ago it was about components, UML, Unified Process (RUP)
- Ten years ago it was about CMMI and Spice
- Four years ago it was about XP
- Yesterday it was about Scrum
- Now it is about Lean and Kanban

All good, but none has all you need!
Software engineering is gravely hampered today by immature practices. Specific problems include:

- The prevalence of fads more typical of fashion industry than of an engineering discipline.
- The lack of a sound, widely accepted theoretical basis.
- The huge number of methods and method variants, with differences little understood and artificially magnified.
- The lack of credible experimental evaluation and validation.
- The split between industry practice and academic research.
We support a process to refound software engineering based on a solid theory, proven principles and best practices that:

- Include a kernel of widely-agreed elements, extensible for specific uses
- Addresses both technology and people issues
- Are supported by industry, academia, researchers and users
- Support extension in the face of changing requirements and technology

This is the Grand Vision
Signatories

- Pekka Abrahamsson,
- Scott Ambler,
- Victor Basili,
- Jean Bézivin,
- Robert V. Binder
- Dines Bjorner,
- Barry Boehm,
- Alan W. Brown,
- Larry Constantine,
- Steve Cook,
- Bill Curtis,
- Donald Firesmith,
- Erich Gamma,
- Carlo Ghezzi,
- Tom Gilb,
- Ellen Gottesdiener,
- Sam Guckenheimer,
- Robert Glass,
- David Harel
- Brian Henderson-Sellers,
- Martin Griss,
- Capers Jones,
- Ivar Jacobson,
- Philippe Kruchten,
- Harold Lawson,
- Robert Martin,
- Bertrand Meyer,
- James Odell,
- Meilir Page-Jones,
- Dieter Rombach,
- Ken Schwaber,
- Alec Sharp,
- Richard Soley,
- Andrey Terekhov
- Fuqing Yang
- Ed Yourdon
Corporate and Academic Signatories

- ABB, Switzerland
- Chalmers, Sweden
- Ericsson, Sweden
- Fujitsu, UK
- Huawei, China
- IBM, USA
- Microsoft, Spain
- KAIST, Korea
- Peking University, China
- KTH Royal Institute of Technology, Sweden
- SAAB, Sweden
- Samsung SDS, Korea
- Swedish Institute of Computer Science, Sweden
- SINTEF, Norway
- Software Engineering Center, Korea
- SEI, USA
- Telecom Italia, Italy
- City of Toronto, Ontario, Canada
- Wellpoint, USA

For current list, please see www.semat.org
Fundamental problem

Everyone of us knows how to develop our software, but as a community we have no widely accepted common ground.
Start from a common ground – The Kernel

- The Kernel should be harvested from a large number of methods.
- The Kernel is practice and method agnostic.
- The Kernel includes elements which are universal for all software development efforts.

The Kernel includes the essence of software engineering

Michelangelo (attributed) “I am freeing the statue from the block”.
Paraphrasing him: “We are freeing the kernel from the methods”.
Extending the Kernel with Practices

- The Kernel is Universal.
- By adding Practices on top of the Kernel you create a specific Method.
- Practices such as Use Cases, Test-Driven Development, Etc
The Kernel provides a shared frame of reference

Many methods all built on the same kernel.
Summing up

Method

Practices

Architecture  Iteration  Use Case  Component

Software Development Kernel

Theory

Analysts

Leaders

Developers

Testers

Practitioners are the target group

SEMAT
Agenda

Semat Presentation

- The Semat Initiative
  - Ivar Jacobson
- The Semat Kernel
  - Mira Kajko-Mattsson
- The Language
  - Michael Goedicke
- Evaluation of Semat
  - Brian Elvesæter
- The Value of Semat
  - Ivar Jacobson

Semat Panel

- Participants
  - The Semat members
  - Bertrand Meyer
  - Barry Boehm
- Subject:
  - Do we need Semat?
The Kernel

- Captures the essence of software engineering
- Forms a map of the software engineering context
- Constitutes a basis for evaluating on-going work
Organizing the Kernel

- Three areas of concern
  - Alphas
  - ....

Customer

Solution

Endeavor
What is an alpha?

• An essential element of the software engineering endeavor that is relevant to an assessment of the progress and health of the endeavor.

• Alpha is an acronym for an Abstract-Level Progress Health Attribute.
The Kernel Alphas

**Customer**
- Opportunity
- Stakeholders

**Solution**
- Requirements
- Software System

**Endeavor**
- Work
- Team
- Way of Working

**Relationships**
- Opportunity < provide > Stakeholders
- Stakeholders < helps to address > Opportunity
- Requirements < demand > Software System
- Software System < fulfils > Requirements
- Requirements scopes and constrains
- Software System supports
- Work < performs and plans > Team
- Team < applies > Way of Working
- Way of Working < guides > Work
- Team < uses and consumes > Software System
- Software System < applies > Stakeholders
- Stakeholders < uses and consumes > Software System

**Set up to address**
- Opportunity

**Additional Notes**
- Customer
- Solution
- Endeavor
The Alpha structure

Checklist
- State
  - XXXXXXXXXXXXXXXXXXXXX
  - XXXXXXXXXXX
  - XXXXXXXXXXXXX
- State
  - XXXXXXXXXXXXXXXXXXXXX
  - XXXXXXXXXXX
  - XXXXXXXXXXXXX
- State
  - XXXXXXXXXXXXXXXXXXXXX
  - XXXXXXXXXXX
  - XXXXXXXXXXXXX
- State
  - XXXXXXXXXXXXXXXXXXXXX
  - XXXXXXXXXXX
  - XXXXXXXXXXXXX
- State
  - XXXXXXXXXXXXXXXXXXXXX
  - XXXXXXXXXXX
  - XXXXXXXXXXXXX
A system made up of software, hardware, and data that provides its primary value by the execution of the software.

A software system can be part of a larger software, hardware, business or social solution.
An architecture has been selected that addresses the key technical risks and any applicable organizational constraints.

An executable version of the system is available that demonstrates the architecture is fit for purpose and supports functional and non-functional requirements.

The system is usable and demonstrates all of the quality characteristics required of an operational system.

The system (as a whole) has been accepted for deployment in a live environment.

The system is in use in a live environment.

The system is no longer supported.
Checklist for Software System

- The criteria to be used when selecting the architecture have been agreed on.
- Hardware platforms have been identified.
- Programming languages and technologies to be used have been selected.
- System boundary is known.
- Significant decisions about the organization of the system have been made.
- Buy, build and reuse decisions have been made.
Using the Kernel in practice

Opportunity

Requirements

Work
Using the Kernel in practice

Opportunity

Requirements

Work

Achieved    In Progress    Not yet achieved
Using the Kernel at KTH Royal Institute of Technology
Agenda

Semat Presentation

- The Semat Initiative
  - Ivar Jacobson

- The Semat Kernel
  - Mira Kajko-Mattsson

- The Language
  - Michael Goedicke

- Evaluation of Semat
  - Brian Elvesæter

- The Value of Semat
  - Ivar Jacobson

Semat Panel

- Participants
  - The Semat members
  - Bertrand Meyer
  - Barry Boehm

- Subject:
  - Do we need Semat?
Motivation

• There are many languages describing software development processes – why a new one?
• Some reasons:
  – Support of a Kernel
  – Support of Dynamic Semantics
  – Focus on Practitioners
A Quick Example

Scrum

- Requirements
- Work
- Team
- Software System
A Quick Example

Support Team

- Sprint Planning Meeting
- Daily Scrum
- Sprint Review
- Sprint Retrospective

Work

Sprint

Sprint Backlog
Some Features of the Language

• Structure and Scalability
  – Layers and Increments

• Extensibility and Flexibility
  – Composition & generic Elements

• Dynamic Semantics
  – Formal basis
• **Meta-levels for static and dynamic semantics**

  • Static semantics defined at meta-level 2: Users define their own types for elements in their endeavor at level 1
  • Dynamic semantics defined at meta-level 1: Providing well-defined functions for instances of types occurring at level 0
Present and Future

- Initial version submitted to OMG
  - Basic structures and mechanisms
  - Plain graphical and textual syntax

- Revised version planned for mid August 2012
  - Improved extension mechanism
  - More completely defined functions in the dynamic semantics
  - Enhanced graphical syntax

- Example library for Practice definitions
Agenda

Semat Presentation

- The Semat Initiative
  - Ivar Jacobson
- The Semat Kernel
  - Mira Kajko-Mattsson
- The Language
  - Michael Goedicke
- Evaluation of Semat
  - Brian Elvesæter

Semat Panel

- Participants
  - The Semat members
  - Bertrand Meyer
  - Barry Boehm
- Subject:
  - Do we need Semat?

The Semat members
- Ivar Jacobson
Evaluation of the SEMAT Kernel and Language

- Exercise the SEMAT Kernel and Language
  - Compare with other approaches (SPEM 2.0 and ISO 24744)

- Illustrate the SEMAT approach
  - Reference examples applying the SEMAT Kernel and Language

- Develop and apply methods for projects
  - Agile development methods composed of practices
List of Practices

- Iterative Development
- Scrum
- User Story
- Use Case
- Test Driven Development
- Concurrent Testing
- Architecture Essentials
- Prince2 Risk Management
Scrum

- Scrum team (roles)
  - Product Owner
  - Development Team (of developers)
  - Scrum Master
- Scrum artifacts
  - Product Backlog
  - Sprint Backlog
  - Increment
- Scrum events
  - The Sprint
  - Sprint Planning Meeting
  - Daily Scrum
  - Sprint Review
  - Sprint Retrospective

- Important note
  - Scrum’s roles, artifacts, events, and rules are immutable and although implementing only parts of Scrum is possible, the result is not Scrum.

- Source
SEMAT Kernel (Alphas)

Opportunity < provides Stakeholders

< helps to address

Requirements < demands Software System

< fulfils

Work < performs and plans Team

< guides

Way of Working < applies
Extending the Work Alpha

• The Work alpha covers the whole duration of a development project that may consist of a number of sprints.

• The Sprint sub-alpha has its own state graph.

• Scrum-specific rules and guidelines can be added as checkpoints for each of the Sprint states.
Enactment using Alpha State Cards

- State cards can be used for reading and understanding the practice, and to progress the states of the Sprint according to the checklist defined.
- Here we show the state card for the Sprint alpha in the Planned state. This requires that all checkpoints are ticked off.
Adding Work Products to the Alphas

- The Sprint Backlog is associated with the Sprint sub-alpha.
- The Product Backlog and Sprint Backlog are associated with the Requirements alpha.
- The Increment is associated with the Software System alpha.
The Scrum Activities define one or more kinds of work items and gives specific guidance on how to perform these.

<table>
<thead>
<tr>
<th>Activity</th>
<th>Alpha Input</th>
<th>Alpha Output</th>
<th>Completion Criterion</th>
<th>Work Product Input</th>
<th>Work Product Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Daily Scrum</td>
<td>Sprint, Team</td>
<td>Sprint</td>
<td>Sprint.Under Control</td>
<td>Sprint Backlog, (Sprint Goal)</td>
<td>Sprint Backlog</td>
</tr>
</tbody>
</table>
Summary

- Kernel
  + Usable map for structuring and defining practices
  + Usable map for selection and composing methods.
    - High-level, abstract view.
      • Value shows first at the sub-alpha level (through practices).
- Language
  + Minimum language focusing on the essentials
    • Not as expressive as SPEM.
    • Where to draw the line?
  ± Graphical syntax.
    – Extension mechanisms and libraries.
- Enactment
  + Support for dynamic semantics
    • Not covered in SPEM.
  + Alpha state cards
    • Appeal to human agents.
    – Alignment with ISO 24744
      • MOF vs Powertype-based metamodelling architectures
- Guidelines and tooling
  – Guidelines for how to use the Kernel and Language
  – Tooling support
    • Initial testing in IBM EPF/RMC
    • IJI Workbench free to the SEMAT community
Agenda

Semat Presentation

- The Semat Initiative
  - Ivar Jacobson
- **The Semat Kernel**
  - Mira Kajko-Mattsson
- **The Language**
  - Michael Goedicke
- Evaluation of Semat
  - Brian Elvesæter
- **The Value of Semat**
  - Ivar Jacobson

Semat Panel

- **Participants**
  - The Semat members
  - Bertrand Meyer
  - Barry Boehm

- **Subject:**
  - Do we need Semat?
Summary - What is new?

- Focus are the practitioners, not the process engineers
- Focus is Method Use and Adaption, not Method Description
- Semat is inclusive and not exclusive – it includes all relevant methods and practices ("good or bad")
- There is a small Kernel of essential
- For small teams and large organizations
- Practices from the bottom, and not process from the top
- Separation of concerns is a fundamental principle
- Light and agile in working with methods
- The old metaphor: ‘Process is program – Team is computer’ is gone
- The process is what the team does. Adaption happens dynamically as a retrospective of what the team did through a feedback loop.

Watts Humphrey (CMMI): “This (SEMAT) meeting in Zurich (2010) is likely to be an historic occasion much like the 1968 NATO session in Garmish.”
What is in it for you as a Software Professional

Your objective is to develop great software quickly.

The kernel helps you to learn, evolve and grow

The kernel helps by providing a firm foundation for your experiences and your team’s way-of-working.

The kernel improves communication allowing for quick learning.

The kernel helps to incrementally improve your way-of-working.
What’s in it for you

Industry
Big companies have many processes.
Challenges:
- Reuse practices
- Reuse training
- “Reuse” of people
- Evolutionary improvement is hard

Software Professional
Want to become experts. Challenges:
- Their skills are not easily reusable for a new product.
- Their career path follows a zig-zag track from hype to hype.

Academics
 Asked to educate and research. Challenges:
- The Gap between research and industry
- No widely accepted theory
- Teaching instances of methods don’t create generalists

SEMAT targets the whole software community.
What you can do now

Become a supporter. Go to www.semat.org
Join the China Chapter of Semat

OMG Proposal Submission

February 22, 2012 In News

We are happy to let you know that on February 20, 2012 the working group of Semat has submitted the proposal, entitled “Essence – Kernel and Language for Software Engineering”, to respond to the OMG RFP: Foundation for the Agile Creation and Enactment of Software Engineering Methods (FACESEM) RFP (OMG Document ad/2011-06-26). The proposal provides comprehensive definitions and descriptions of the kernel and the language of software engineering methods, which meet (address) the mandatory requirements set forth in the FACESEM RFP.

This proposal can be found here: “Essence – Kernel and Language for Software Engineering”.

The working group has been working intensively for the past a couple of months in preparing for this submission. The next step is to give a presentation during an OMG meeting in March in Washington D.C. There are 3 initial submissions which will all present during the OMG meeting.
What you can do now

Become a supporter. Go to www.semat.org
Join the China Chapter of Semat

Read the OMG submission
What you can do now

- Become a supporter. Go to www.semav.org
- Join the China Chapter of SEMAT
- Read the OMG submission
- Read the new book: "The Essence of Software Engineering" by Ivar Jacobson, Pan-Wen Ng, Paul D. McMahan, Ian Spence, Swante Lidman.
What you can do now

- Become a supporter. Go to www.semat.org
- Join the China Chapter of Semat
- Read the OMG submission
- Read the new book
- Join Us!!!
References

References


You can find these and more on www.semat.org
Agenda

Semat Presentation

- The Semat Initiative
  - Ivar Jacobson
- The Semat Kernel
  - Mira Kajko-Mattsson
- The Language
  - Michael Goedicke
- Evaluation of Semat
  - Brian Elvesæter
- The Value of Semat
  - Ivar Jacobson

Semat Panel

- Participants
  - The Semat members
  - Bertrand Meyer
  - Barry Boehm
- Subject:
  - Do we need Semat?
Barry Boehm’s slides
Why Do We Need SEMAT?

• Its principles are needed for most future software systems
  – Stakeholder value-based; balancing requirements and architecture; attacking the risks; working incrementally; lean/agile emphasis

• It balances practice checklists and freedom to innovate
  – Alpha states and activities

• It has good case studies of usage
  – Low entry barrier for usage

• It will grow even stronger with broad use and feedback
  – Complex, high assurance systems of systems
  – Avoiding agile failure modes
Bertrand Meyer’s slides
SEMAT kernel idea

1. **Universals**
2. **Practices**
3. **Methods**

**The kernel**

- Composed of
- Defined in terms of

**Kernel language**

**Patterns**

**Universals**
Varieties of software engineering work

- Conceptual
- Constructive
- Analytic
- Empirical
What should Semat be doing?

- Conceptual
  - Pinpoint the most powerful concepts in software engineering

- Constructive
  - Build models of methods, practices, patterns

- Analytic

- Empirical
  - Say what works and what does not
An example of formal description

- **class** METHOD **feature**
  - practices: SET [PRACTICES]
  - ...
- **end**
- **class** PRACTICE **feature**
  - task: TASK
  - period (p: PROJECT): PERIODICITY
- **end**
- **class** PROJECT **feature**
  - method: METHOD
  - deadlines: PERIODICITY
  - **invariant**
    - **across** method.practices **as** mp **all**
    - mp.period (**Current**).compatible (deadlines)
- **end**
An example of empirical/analytical work

AGILE!

The good, The hype & The ugly

Bertrand Meyer